Attorney Docket No. 04906.P078
Express Mail No. EL 485755408US

UNITED STATES PATENT APPLICATION

FOR

METHOD AND APPARATUS FOR PROCESS SYNC RESTART

Inventors:

Ravi Chandra
Gerald Neufeld
Jenny Yuan

Prepared by:

Blakely, Sokoloff, Taylor & Zafman LLP
12400 Wilshire Blvd., Suite 700
Los Angeles, California 90025
(310)207-3800

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number: **EL 485755408US** Date of Deposit: **June 2, 2001**

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Commissioner of Patents and Trademarks, Washington, D. C. 20231

Virginia Velazquez
(Typed or printed name of person mailing paper or fee)

Virginia Velazquez
(Signature of person mailing paper or fee)

6-02-01
(Date signed)

Method and Apparatus for Process Sync Restart

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] The invention relates to the field of networks. More specifically, the invention relates to network elements.

Background of the Invention

[0002] A network element hosts multiple processes to maintain data for network communication. These processes relay information to each other with inter-process communication (IPC). The middleware of the network element will maintain process identification numbers for the processes running on the network element. One process will communicate directly with another process using these process identification numbers. Often within a network element, multiple processors run different operating systems.

[0003] If a process wants to communicate with a process that is dead, the process continues passing requests to the dead process. The requesting process detects the failure of the dead process through a response to the request or via timeouts. Although the operating system can detect when a process dies, it does not immediately communicate state of the process to other processes.

[0004] One method of IPC utilizes heartbeat messaging between processes. Once communication is established between two processes on a network element, the two processes periodically transmit heartbeat messages or signals indicating that they are alive and running. Death of one of the processes is detected by the other process when a heartbeat message has not been received within a given time period. Once a process

is dead, however, the living process is ignorant of the dead process restarting. In addition, if both communicating processes die, when they restart different scenarios can occur. If both processes restart within the same time period, then they will both send requests. If one process restarts while the other remains dead, then the requesting process will repeatedly transmit requests to the dead process until it restarts.

[0005] Processes communicate with each other to disseminate information. One process on a network element may gather information about the interfaces of the network element while another process gathers routing information. This information is exchanged and/or passed on to other processes to facilitate processing and transmission of network traffic.

[0006] When a process requires information from another process, the process will send an IPC message to the other process requesting information or data. The other process will then pass a response back to the requesting process with the requested data.

[0007] If a requesting process does not receive a response within a certain time period, then the requesting process will mark the data from the timed out process as stale. Since the requesting process is unaware of the state of the timed out process, it sets a long timer on the stale data. When the timer expires, the stale data is removed.

[0008] Unfortunately, without information about the state of the timed out process, the requesting process cannot function intelligently. The stale data may be used beyond its life. Traffic processed with the stale data may be dropped or delayed. The length of time the data should be considered stale begins at some point before the timeout until the time expires. The amount of traffic impacted increases in proportion to this length of time.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] The invention may best be understood by referring to the following description and accompanying drawings that are used to illustrate embodiments of the invention. In the drawings:

[0010] Figure 1 is an exemplary diagram illustrating inter-process communication according to one embodiment of the invention.

[0011] Figure 2 is an exemplary diagram illustrating inter-process communication among network elements according to one embodiment of the invention.

[0012] Figure 3 is a flowchart for a process to register with the process state manager according to one embodiment of the invention.

[0013] Figure 4 is a flowchart for performing a lookup request according to one embodiment of the invention.

[0014] Figure 5 is a flowchart for the process state manager to process requests according to one embodiment of the invention.

[0015] Figure 6 is a flowchart for processing a lookup request according to one embodiment of the invention.

[0016] Figure 7A is a flowchart for determining death of a process on the same network element as the PSM according to one embodiment of the invention.

[0017] Figure 7B is a flowchart for the process state manager to process heartbeat messages according to one embodiment of the invention.

[0018] Figure 7C is a flowchart for the process state manager to determine death of a process according to one embodiment of the invention.

[0019] Figure 8 is a flowchart for attempting inter-process communication according to one embodiment of the invention.

[0020] Figure 9 is an exemplary diagram of process interaction according to one embodiment of the invention.

[0021] Figure 10 is a diagram of the processes illustrated in Figure 9 and their locations in memory according to one embodiment of the invention.

[0022] Figure 11 is a flowchart for limiting stale data according to one embodiment of the invention.

[0023] Figure 12 is a flowchart of initialization for a restarted process according to one embodiment of the invention.

[0024] Figure 13 is a flowchart for synchronization of data according to one embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

[0025] In the following description, numerous specific details are set forth to provide a thorough understanding of the invention. However, it is understood that the invention may be practiced without these specific details. In other instances, well-known circuits, structures and techniques have not been shown in detail in order not to obscure the invention.

<u>The Process State Manager</u>

[0026] Figure 1 is an exemplary diagram illustrating inter-process communication according to one embodiment of the invention. In Figure 1, a process state manager (PSM) 101 communicates with a process 103 and a process 105. The PSM 101 provides communication keys to the processes 103 and 105 when they register with the PSM 101. With the communication keys, the processes 103 and 105 communicate with each other. The process 103 requests a communication key from the PSM 101 whenever the process 103 starts and restarts. In general, process 103 and 105 exist on different processors using different operating systems. The process 105 also requests a communication key from the PSM 101 whenever it starts and restarts.

[0027] Assuming the process 103 is interested in the process 105, the process 103 registers interest in the process 105 with the PSM 101 by sending a lookup request to the PSM 101. The lookup request can identify the process 105 by a symbolic name, an

identifier provided by the process 105. After the process 105 has registered with the PSM 101 and the process 103 registers interest of process 105 with the PSM 101, the PSM 101 passes process 105's communication key to process 103. If the process 105 has not registered with the PSM 101, then the process 103 waits until the process 105 registers with the PSM 101 or polls the PSM 101 for the process 105's communication key. The process 103 uses the communication key for inter-process communication (IPC) with the process 105. The process 105 will compare the communication key transmitted by process 103 with its communication key. If the keys do not match, then process 105 rejects messages from process 103.

[0028] The communication key includes a process identifier and an incarnation identifier. The process identifier is unique for each process registered with the PSM 101. The unique process identifier identifies a process. The incarnation identifier indicates an incarnation or version of the process. When a process first starts, its incarnation identifier is an initial value. Each time the process restarts, its incarnation identifier is updated to reflect the new version or incarnation.

[0029] Figure 2 is an exemplary diagram illustrating inter-process communication between processors according to one embodiment of the invention. In Figure 2, a processor 201 hosts the PSM 101 and the process 103. The process 103 communicates with the process 105. The process 105 is running on a processor 203. The process 105 registers and communicates with the PSM 101. The process 105 also communicates with the process 103. Since the process 105 is on the non-PSM processor 203, the process 105 transmits signals indicating that it is running (i.e. heartbeat messages or breath of life messages). If the PSM 101 does not receive a heartbeat message within a defined time period, then the PSM 101 considers the process as dead.

[0030] For example, if the process 105 dies, then it will no longer transmit heartbeat messages to the PSM 101. The PSM marks the process 105 as dead when the defined time period for receiving a heartbeat message from the process 105 expires. Since the processes 103 is interested in the process 105, the PSM 101 will transmit a death notification to the interested process 103. With this information, the process 103 can function intelligently and perform other tasks without expending time attempting communication with the dead process 105. When the process 105 restarts, it will request a communication key from the PSM 101. The PSM 101 will find the already created process identifier for process 105 and update the incarnation identifier for the process 105 to indicate the new incarnation. After updating the incarnation identifier, the PSM 101 transmits the new communication key to the process 105. The PSM 101 then transmits the new communication key for process 103 to the interested process 103. Process 103 receives the new communication key for process 105 asynchronously. Once received, the process 103 can begin communication with process 105.

[0031] In one embodiment of the invention, the process 103 also transmits heartbeat messages to the PSM 101. In another embodiment of the invention, an operating system running on the processor 201 determines when the process 103 dies. If the process 103 dies, then the operating system will notify the PSM 101 of process 103 dying.

[0032] The described embodiments of the invention provide intelligence to processes. Processes can efficiently and intelligently perform tasks with knowledge of which processes are available for communication. A process does not expend time attempting to establish communications with a dead process. Instead, the process can complete other tasks until the dead process restarts.

[0033] Figure 3 is a flowchart for a process to register with the process state manager according to one embodiment of the invention. At block 301, a process starts. At block 303, the process transmits a register request to the PSM. At block 305, the process receives a communication key from the PSM. At block 307, the process begins to periodically transmit heartbeat messages to the PSM. In one embodiment of the invention, the process runs on the same processor as the PSM and does not transmit heartbeat messages.

[0034] Figure 4 is a flowchart for performing a lookup request according to one embodiment of the invention. At block 401, a process transmits a lookup request of a process to the PSM. At block 403, the process determines if the PSM returns a communication key for the requested process. If the PSM returns a communication key to the requesting process, then at block 405 the requesting process uses the key to communicate with the requested process. If a communication key is not returned by the PSM at block 403, then at block 407 the requesting process performs other functions while waiting for the PSM to transmit the requested communication key. In another embodiment of the invention, the requesting process polls the PSM until the requested key is received.

[0035] Figure 5 is a flowchart for the process state manager to process requests according to one embodiment of the invention. At block 501, the PSM receives a request. At block 503, the PSM determines if the received request is a register request. If the received request is not a register request, then at block 509 the lookup request is processed.

[0036] If the PSM determines the received request to be a register request at block 503, then at block 511 the PSM determines if the requesting process was dead. If the requesting process was not dead (i.e., the requesting process has previously registered,

then the PSM creates a unique process identifier and a new incarnation identifier for a communication key at block 515. Control flows from block 515 to block 517. If the requesting process was dead, then at block 513 the PSM uses the process identifier for the requesting process and updates the requesting process' incarnation identifier to create a new communication key. At block 517, the PSM transmits the communication key to the requesting process. At block 519, the PSM transmits a birth notification indicating the new communication key to processes interested in the requesting process.

[0037] Figure 6 is a flowchart for processing a lookup request indicated in block 509 of Figure 5 according to one embodiment of the invention. At block 601, the PSM performs a lookup of the requested process. In one embodiment of the invention, the lookup request includes the string name of the requested process. In another embodiment of the invention, the lookup request includes an identifier for the process provided by the operating system in combination with a value identifying the hosting network element. At block 603, it is determined if the requested process has registered with the PSM. If the requested process has not registered with the PSM, then at block 605 the PSM notes the requesting process as an interested process of the requested process. If the PSM finds the requested process, then at block 607 the PSM registers the requesting process as an interested process for the requested process. At block 609, the PSM determines if the requested process is alive. If the process is alive, then at block 611 the PSM transmits the communication key for the requested process to the requesting process. If the requested process is dead, then at block 613, the PSM transmits a death notification to the requesting process.

[0038] Figures 7A-7C are flowcharts for the PSM to determine death of a process according to one embodiment of the invention. Figure 7A is a flowchart for determining death of a process on the same network element as the PSM according to

one embodiment of the invention. At block 701, the PSM uses the operating system to determine death of a process. In one embodiment of the invention, the PSM watches for the operating system to generate error codes for processes. The PSM determines which process has died from the error code. In another embodiment of the invention, the PSM periodically queries the operating system for currently active processes. The PSM determines processes to be dead when they are no longer listed as active by the operating system. At block 703, the PSM updates the state of the process to indicate death. At block 705, the PSM transmits a death notification to the interested processes registered for the dead process.

[0039] Figure 7B is a flowchart for the process state manager to process heartbeat messages according to one embodiment of the invention. At block 707, the PSM receives a heartbeat message. At block 708, the PSM determines which process transmitted the heartbeat message. At block 709, the PSM resets a counter for the transmitting process.

[0040] Figure 7C is a flowchart for the process state manager to determine death of a process according to one embodiment of the invention. At block 711, the PSM initializes a counter for a registering process. At block 713, the PSM increments the counter. At block 715, the PSM determines if the counter has exceeded a limit for receiving heartbeats from the process. If the limit has not been exceeded, then control flows back to block 715. If the limit has been exceeded, then at block 717 the PSM updates the state of the process to indicate dead. At block 719, the PSM transmits a death notification to processes interested in the dead process. In another embodiment of the invention, the PSM transmits a message to a process exceeding the time limit. If the process responds, then the counter is reset as if a heartbeat message has been received.

[0041] Figure 8 is a flowchart for inter-process communication according to one embodiment of the invention. At block 801, a process receives an IPC message. At block 803, the receiving process determines if the communication key included in the IPC message matches the receiving process' communication key. If the keys match, then at block 805, the receiving process accepts communication with the transmitting process. If the keys do not match, then the receiving process rejects communications from the transmitting process at block 807. In one embodiment of the invention, the transmitting process transmits a lookup request to the PSM in response to the rejected communication.

[0042] As stated above, the described embodiments of the invention provide intelligence to processes. With this intelligence, processes can performs tasks efficiently. In addition, the incarnation identifier of the communication key provides intelligence of a process restarting. A process may have a different task set upon determining another communication has restarted. For example, an interested process may request a refresh of data from the restarted process.

Process Sync Restart

[0043] Figure 9 is an exemplary diagram of process interaction according to one embodiment of the invention. In Figure 9, a configuration manager 905 communicates with 3 network processes 901, 903, and 909. The configuration manager 905 sends configuration information to the network processes 901, 903, and 909. In the example illustrated by Figure 9, the network process 901 communicates with the network process 903. The network process 903 communicates with the network process 907. The network process 909 also communicates with the network process 907. Each of the network processes 901, 903, 907, and 909 gather or discover information and generate data corresponding to the discovered information. For example, if the network

process 903 is an interface state manager, then the network process 903 would discover the states of the interfaces of the hosting network element (e.g., up, down, cable connected, etc.) and communicate those states to other network processes. If the network process 903 is a Border Gateway Protocol (BGP) process, then the network process 903 would gather routing information and communicate that information to other network processes, such as the network process 907.

[0044] As an illustration, assume the network process 901 discovers 3 interfaces on its host network element: interface 1, interface 2, and interface 3. The network process 901 determines that all 3 interfaces are up and have cables connected. The network process 901 communicates this information to the network process 903. The network processes 903 stores this information from the network process 901 and uses it to determine routing information. The network process 903 determines the routing information as indicated in Table 1.

Destination Address	**Interface**
1.1.1.1	1
2.2.2.2	2
3.3.3.3	3

Table 1: Exemplary Routing Information

[0045] The network process 901 dies and restarts. The network process 901 discovers that the interface 2 is down, but discovers an interface 4 is up and has a cable connected. The network process 901 communicates this new information to the network process 903. The network process 903 synchronizes this new information with the stored information. The network process 903 then modifies its routing information as indicated in Table 2.

Destination Address	Interface
1.1.1.1	1
3.3.3.3	3
4.4.4.4	4

Table 2: Updated Exemplary Routing Information

[0046] The change in information ripples through the communicating network processes. The network process 907 previously stored the information shown in Table 1. The network process 907 will receive the information shown in Table 2 from the network process 903. When the network process 907 synchronizes the two sets of data, the absence of information for 2.2.2.2 implies that it should be removed. The network process 909 transmits data using the information from the network process 907. Although some traffic transmitted to 2.2.2.2 may be lost because the interface 2 goes down, the network process 909 can still transmit traffic to 1.1.1.1 and 3.3.3.3 despite the death of the network process 901. In addition, if the network process 903 dies, the network process can still transmit traffic without interruption. The described mechanism for seamlessly synchronizing data from restarted processes avoids service delay and service interruption typically caused by internal errors. Hence, the described invention increases robustness and reliability of a network element.

[0047] Figure 10 is a diagram of the processes illustrated in Figure 9 and their locations in memory according to one embodiment of the invention. Although the memory area of the memory 1002 for each process is equal in Figure 10, each process may use or be provisioned a different amount of memory. Furthermore, each of the areas of memory 1001, 1003, 1007, and 1009 are shown as a single segment of the memory 1002, but multiple words or segments of the memory 1002 may comprise each area. In Figure

10, the network processes 901, 903, 907, and 909 each use respectively the areas of memory 1001, 1003, 1007, and 1009.

Referring to the example described above, the information gathered by the network process 901 is stored in its memory area 1001. The information gathered by the network process 903 is stored in the area of memory 1003. Since the network process 903 has requested information from the network process 901, information gathered by the network process 901 is also stored in the area of memory 1001. The network process 907 stores information from the network process 903 in the area of memory 1007. Therefore, referring the to above described example, interface information is stored in the areas of memory 1001, 1003 and 1007. If the network process 909 requests interface information, then interface information will also be stored in the area of memory 1009. Routing information collected by the network process 903 will be stored in the area of memory 1001, 1003, and possibly 1009 if the network process 909 requests such information from the network process 903 directly or via the network process 907. In another embodiment of the invention, the memory 1002 is multiple memories.

[0048] Figure 11 is a flowchart for limiting stale data according to one embodiment of the invention. Figure 11 will be described with reference to the previously described example and Figure 9. At block 1101, the network process 903 receives a death notification for the network process 901. At block 1103, the network process 903 initializes a timer. At block 1105, the network process 903 indicates all data from the network process 901 as stale. At block 1107, the network process 903 determines if the timer is greater than or equal to a time limit. If the time has expired, then at block 1109, the network process 903 clears stale data from the network process 901 and any new data received from the network process 901. If the time has not expired, then at

block 1111 the timer is incremented and control flows back to block 1107. Limiting the life of stale data prevents magnifying effects of data that may be causing the originating process to repeatedly die or loop.

[0049] Figure 12 is a flowchart of initialization for a restarted process according to one embodiment of the invention. Figure 12 will be described with reference to the previously described example and Figure 9. At block 1201, the dead network process 901 is restarted. At block 1203, the network process 901 gets configurations from the configuration manager 905. At block 1205, the network process initializes data (i.e., discovers state of interfaces). At block 1207, the network process 901 determines if it has completed initialization. If the network process 901 has completed initialization, then at block 1209 the network process 901 transmits an EOF or signal indicating completion (done signal) to the network process 903. If the initialization is not complete, then at block 1211 it is determined if the process has died again. If the process has not died again, then control flows to block 1207. If the network process 901 has died again, then control loops back to block 1201.

[0050] Figure 13 is a flowchart for synchronization of data according to one embodiment of the invention. Figure 13 will be described with reference to the previously described example and Figure 9. At block 1301, the network process 903 receives data from the network process 901. At block 1303, the network process 903 determines if data it currently has from the network process 901 is stale. If the current data is not stale, then at block 1305 the network process 903 updates the network process 901 data. If the data is marked as stale, then at block 1307 the network process 903 stores the received data as temporary data. At block 1309, the network process 903 determines if it has received an EOF or done signal from the network process 901. If the network process 903 has not received the EOF, then control loops back to block

1301. If the network process 903 receives the EOF from the network process 901, then at block 1311 the network process 903 stops incrementing the timer corresponding to the network process 901. At block 1313, the network process 903 synchronizes the temporary data with the stale data and clears the stale indicator.

[0051] The described embodiments of the invention improve reliability of a network element. Providing intelligence to the processes of a network element enables processes to function efficiently as previously stated. In addition, intelligence about other processes enables processes of a network element to function independently despite failures without interrupting service. Each process can use stored data from other processes to facilitate processing and/or transmission of traffic even though other processes are dead. Knowledge of other process' states also enable processes to determine how long data can be used and if the data can be refreshed.

[0052] The described network elements include line cards and control cards executing the described processes. The line cards and control cards of the network elements include memories, processors, and/or Application Specific Integrated Circuits ("ASICs"). Such memory includes a machine-readable medium on which is stored a set of instructions (i.e., software) embodying anyone, or all, of the methodologies described herein. Software can reside, completely or at least partially, within this memory and/or within the processor and/or ASICs. For the purpose of this specification, the term "machine-readable medium" shall be taken to include any mechanism that provides (i.e., stores and/or transmits) information in a form readable by a machine (e.g., a computer). For example, a machine-readable medium includes read only memory ("ROM"), random access memory ("RAM"), magnetic disk storage media, optical storage media, flash memory devices, electrical, optical, acoustical, or

other form of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.), etc.

[0053] While the invention has been described in terms of several embodiments, those skilled in the art will recognize that the invention is not limited to the embodiments described. The method and apparatus of the invention can be practiced with modification and alteration within the spirit and scope of the appended claims. The description is thus to be regarded as illustrative instead of limiting on the invention.